**复杂度事后分析报告**

学生姓名： 黄昌盛 学 号： 6109118110 专业班级： 计科软件183班

实验类型：□ 验证 □ 综合 □ 设计 □ 创新 实验日期： 2021.3.17 实验成绩：

**一、题目描述和要求**

货郎担问题是这样一个问题：给定一系列城市和每对城市之间的距离，求解访问每一座城市一次并回到起始城市的最短回路。它是组合优化中的一个NP困难问题，在运筹学和理论计算机科学中非常重要。

以教材p6算法1.4的货郎担问题为例，进行事后复杂度分析，最后写一个报告。

**二、报告步骤**

1、运行时间提取

利用C++的clock()函数，在进行调用解决货郎担问题的函数之前之后，分别记录算法的开始时间和结束时间，然后用结束时间减去开始时间乘1000与是1秒内CPU运行的周期数CLOCKS\_PER\_SEC做除法得到以毫秒为单位的运行时间。

|  |
| --- |
| 1. **clock\_t** start = clock(), end; 2. salesman\_problem(n, mincost, path, c); // 计算最小花费函数 3. end = clock(); 4. // 输出运行时间 5. cout << "n" << " = " << n << "  " 6. << **int**(end - start) \* 1000 / CLOCKS\_PER\_SEC << "ms" << endl; |

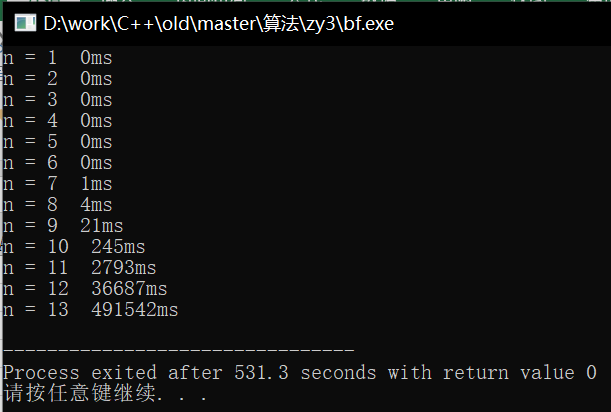
2、随机数据生成

利用系统当前时间做种子，使用C++11 中新加入的特性mt19937 产生伪机数,获得费用矩阵的随机数据。

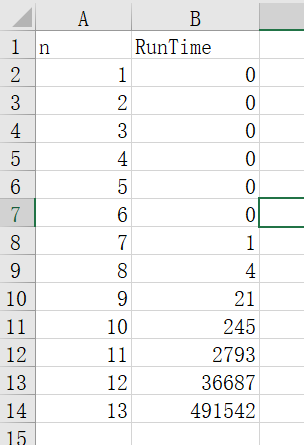
|  |
| --- |
| 1. unsigned seed = std::chrono::high\_resolution\_clock::now() 2. .time\_since\_epoch().count(); 3. std::mt19937 mt(seed); 4. // 随机数据生成, 存入费用矩阵 5. **for** (**int** i = 0; i < n; i++) 6. **for** (**int** j = 0; j < n; j++) 7. c[i][j] = 100 + mt() % 100; |

3、自动输出批量的不同规模及其运行时间数据

输出规模n，和对应运行时间t，运行结果如下：

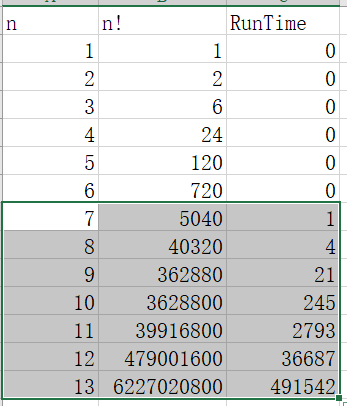


4、输出数据放入Excel

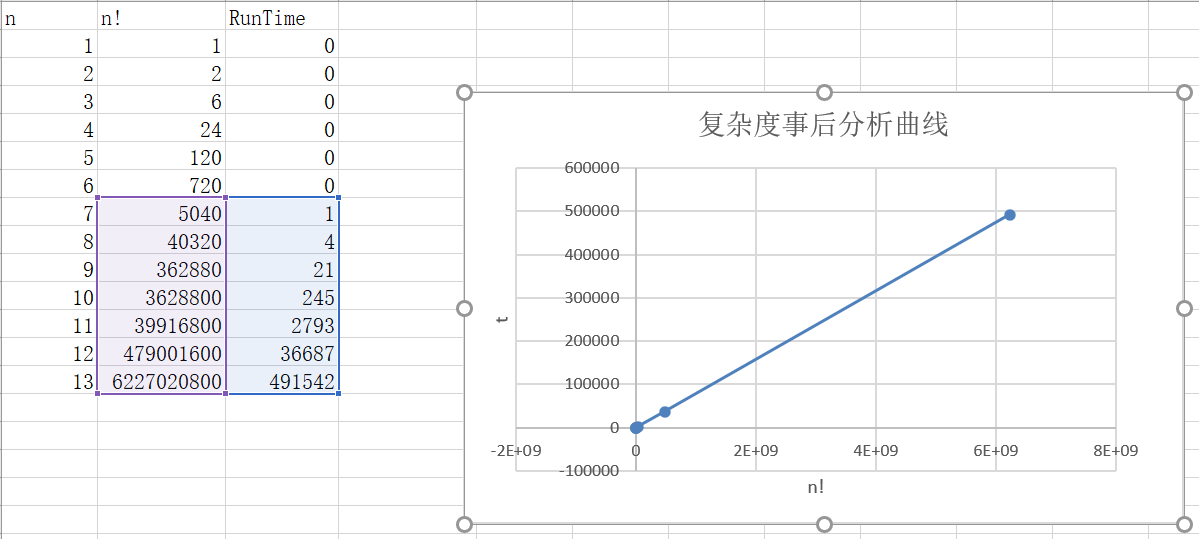


5、Excel数据表格及曲线生成

使用Excel计算出n的阶乘，这里注意要排除时间为0的点，使用n!和时间t绘制曲线：

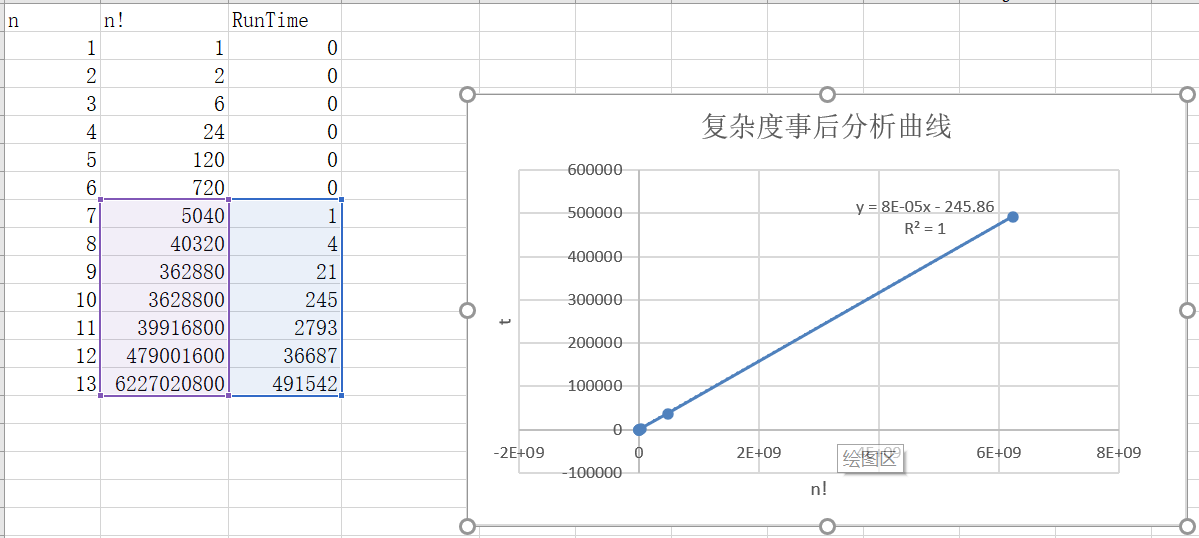


观察分析图像发现两者存在线性关系。



6、EXcel趋势线的生成及复杂度的提取

采用n!和运行时间t进行拟合



由结果可知，t = 8E-5\*n!，R^2 = 1,该算法的时间复杂度为O(n!)

7、算法的分析

由程序运行结果可知，虽然穷举法很简单易懂，对个城市枚举一个的全排列，在其中找到最优解，这样确实能够找到最优解，但是复杂度达到(n-1)!，当城市数量很小时，运行时间是很小，但当城市数量大于12个时，我们就不能够在短时间内求解，n=13时我的电脑跑了8分多钟，显然这不是一个好的方法。可以考虑其他如状压dp的优化算法等。

**三、程序代码**

|  |
| --- |
| 1. #include <bits/stdc++.h> 2. **using** **namespace** std; 3. **const** **int** INF = 0x3f3f3f3f; 4. unsigned seed = std::chrono::high\_resolution\_clock::now() 5. .time\_since\_epoch().count(); 6. std::mt19937 mt(seed); 7. **int** t, n, path[100]; 9. **void** salesman\_problem(**int** n, **double** &min, **int** t[], vector<vector<**int**>> c) { 10. **int** p[n], i = 1; 11. **double** cost; 12. **for** (i = 0; i < n; i++) p[i] = i; 14. min = INF; 15. **do** { 16. cost = 0; 17. **for** (**int** i = 0; i < n - 1; i ++) 18. cost += c[p[i]][p[i + 1]]; 19. cost += c[p[n - 1]][p[0]]; 20. **if** (cost < min) { 21. **for** (i = 0; i < n; i++) 22. t[i] = p[i]; 23. min = cost; 25. // 打印当前最优解 26. //          cout << min << endl; 27. //          for (int i = 0; i < n; i++) { 28. //              cout << t[i] << "->"; 29. //          } 30. //          cout << t[0] << endl; 31. } 32. } **while** (next\_permutation(p, p + n)); 34. } 36. **int** main() { 37. **while** (n < 13) { 38. n++; 39. vector<vector<**int**> > c(n, vector<**int**> (n)); 40. **double** mincost = 0; 41. // 随机数据生成, 存入费用矩阵 42. **for** (**int** i = 0; i < n; i++) 43. **for** (**int** j = 0; j < n; j++) 44. c[i][j] = 100 + mt() % 100; 46. **clock\_t** start = clock(), end; 47. salesman\_problem(n, mincost, path, c); // 计算最小花费函数 48. end = clock(); 49. // 输出运行时间 50. cout << "n" << " = " << n << "  " 51. << **int**(end - start) \* 1000 / CLOCKS\_PER\_SEC << "ms" << endl; 53. // 打印答案 54. //      cout << mincost << endl; 55. //      for (int i = 0; i < n; i++) { 56. //          cout << path[i] << "->"; 57. //      } 58. //      cout << path[0] << endl; 59. } 60. } |